

Development of physics-informed neural networks for solving partial differential equations

Dmytro Alianakh¹, Ivan Dyyak¹ and Ihor Makar¹

¹ Ivan Franko National University of Lviv, 1 Universytetska Street, Lviv, 79000, Ukraine

Abstract

This study evaluates the effectiveness of physics-informed neural networks (PINNs) for solving both stationary and non-stationary partial differential equations (PDEs), including those with Robin boundary conditions, in rectangular and non-rectangular domains. Although only a small subset of the PINN literature examines mixed boundaries or tackles non-rectangular geometries, and even fewer studies benchmark accuracy against the finite-element method (FEM), the present work provides precisely that comparison. We test feedforward PINNs with \tanh activation, whose depth and width were empirically selected for each benchmark to balance accuracy and training cost; training uses Adam optimization with Glorot initialization. These networks are evaluated on three problems: a 2-D Laplace equation on a square (Dirichlet–Neumann–Robin), the same equation on a doubly connected domain with Dirichlet boundary conditions, and a 1-D non-stationary heat equation with Robin boundaries. A weighted mean-squared residual, evaluated via automatic differentiation in TensorFlow, balances equation, boundary, and initial-time terms, thereby handling non-stationary problems without a separate time-stepping scheme. Within the tested class, linear, second-order parabolic and elliptic PDEs in 1-D and 2-D, the network attains $\leq 3\%$ L_∞ error relative to analytical or FEM solutions after 4–6 min of training on an RTX 3080 Ti Laptop GPU, matching FEM accuracy while eliminating meshing and easing equation and boundary changes. The time to compute a standard PINN solution is longer than for a FEM solution for problems considered in the research, and a broad literature review reveals theoretical convergence limits that constrain standard PINNs to modest-scale, well-conditioned diffusion problems.

Keywords

physics-informed learning, PINNs, PDE, automatic differentiation, FEM, boundary conditions

1. Introduction

In the contemporary scientific community, significant attention is dedicated to developing numerical methods for solving differential equations, which have applications across various scientific and engineering fields. A leading technology in this domain is physics-informed neural networks (PINNs) [1, 2, 3], which integrate physical laws directly into the neural network training process [4], ensuring high accuracy and efficiency in solving complex problems. This relies on the universal approximation theorem: any continuous function on a compact set can be approximated arbitrarily well by a sufficiently large multilayer feedforward network with a non-polynomial activation [5].

PINNs have undergone significant advancements, enabling them to solve a broader spectrum of partial differential equations (PDEs), including one-dimensional, nonlinear, and two-dimensional stationary problems [6, 7, 8]. The framework presented in [9] facilitates both forward and inverse problem-solving involving nonlinear PDEs. Recent modifications of PINNs include VPINNs [10], which reformulate the loss function using a variational (weak) formulation to improve robustness, KANN-based models [11] that leverage Kolmogorov–Arnold Networks for enhanced parameter efficiency and faster convergence, and others. While they demonstrate impressive examples, several gaps persist: (i) mixed Robin conditions are rarely tested. To our knowledge, only five studies [12, 13, 14, 15, 16] explicitly address Robin boundary conditions in PINNs, even though the overall PINN literature numbers in the hundreds. Moreover, those works focus on specialized PINN variants rather than the standard PINN formulation employed here; (ii) comparison usually is done with an analytical solution, and FEM is usually absent. Only one of the articles mentioned compares PINN with FEM. To

¹CMIS-2025: Eighth International Workshop on Computer Modeling and Intelligent Systems, May 5, 2025, Zaporizhzhia, Ukraine

✉ dmytro.alianakh@lnu.edu.ua (A. Alianakh); ivan.dyyak@lnu.edu.ua (I. Dyyak); ihor.makar@lnu.edu.ua (I. Makar)



0009-0009-3482-3383 (A. Alianakh); 0000-0001-5841-2604 (I. Dyyak); 0009-0009-7590-3865 (I. Makar)



© 2025 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

our knowledge, the literature still lacks a systematic PINN vs FEM comparison for equations of diffusion type, such as the 2-D Laplace equation and the 1-D heat equation, when mixed Dirichlet–Neumann–Robin conditions are imposed on rectangular domains or when purely Dirichlet boundary conditions are prescribed on non-rectangular geometries. Our work closes these gaps by providing a comparison of [Примітки] diffusion and heat-conduction problems with analytically known or FEM solutions. Also, some papers like [1, 2] use training datasets generated from another solver, while in our approach we only have physics loss.

This research is driven by the need for methods capable of solving physical problems with high accuracy and low computational costs. Traditional numerical methods, such as the finite element method (FEM) [17, 18], often require substantial computational resources and time, especially for multidimensional systems, nonlinear problems, and heterogeneous media.

The goal of this study is to determine the accuracy and computational cost trade-offs under which a mesh-free PINN can replace FEM for linear, second-order parabolic and elliptic PDEs. We demonstrate this on stationary 2-D problems, rectangular domains with mixed boundary conditions and non-rectangular domains with Dirichlet boundary conditions, and on a 1-D non-stationary heat-conduction problem, testing PINN predictions against analytical and FEM solutions.

2. Methodology

A physics-informed neural network is a feedforward neural network that incorporates the laws of physics, which are defined by differential equations, into the learning process. Figure 1 shows a PINN architecture with an input layer, hidden layers, an output layer and components of a loss function.

In our case, the algorithm of this approach can be presented as follows:

- Set datasets for training: for the equation, for the boundary and initial conditions;
- Define the calculation of the necessary derivatives for the equations that are used in the loss function;
- Define the loss function for the equation and boundary conditions;
- Train the PINN to find an approximate solution by minimising the loss function.

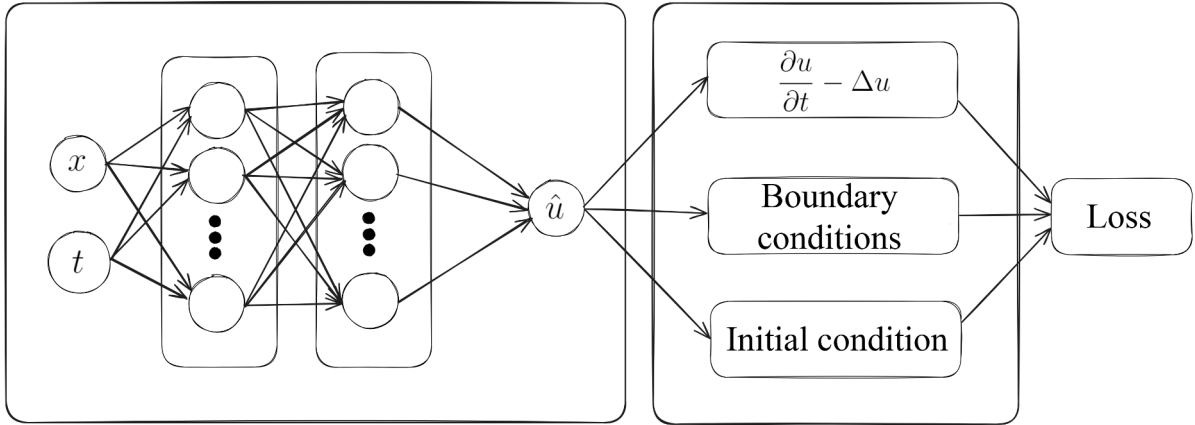


Figure 1: Architecture of a physics-informed neural network for solving the non-stationary heat equation.

It is assumed that all the necessary input data are initially set, after which the neural network is trained, and the values at specific points are predicted. If any of the input parameters are modified, the neural network must be retrained.

The loss function is calculated as the sum of the mean squared errors (MSE) of the equation values, initial and/or boundary conditions.

The MSE is calculated according to formula (1).

$$MSE(e) = \frac{1}{n} \sum_{i=1}^n e_i^2, \quad (1)$$

where e is a vector of length n (n is the number of training points) obtained after substituting the obtained approximation from the neural network into the equations or initial/boundary conditions.

The loss function is:

$$L = \lambda_e MSE_e + \lambda_b MSE_b + \lambda_i MSE_i, \quad (2)$$

where MSE_e is the error of satisfaction of the equation, MSE_b is the error of satisfaction of the boundary conditions, MSE_i is the error of satisfaction of the initial condition, λ_e , λ_b and λ_i are weighting factors to balance the influence of different components of the loss function. The magnitudes of residuals of PDE, boundary conditions and initial condition can differ by orders of magnitude, one term may dominate the total loss, preventing the network from accurately enforcing the other conditions; this imbalance motivates careful choice of the weights.

The physics-informed neural networks were developed using Python and TensorFlow (TF) version 2. TensorFlow's automatic differentiation mechanism [19] was used to calculate the values of the loss function and determine all the necessary derivatives, both from the known and the desired function. The Adam (adaptive moment estimation) algorithm [20], a stochastic gradient descent method based on estimates of the first and second moments of gradients, was used to optimize the model's parameters. To initialize the weights, we used the Glorot Normal initialization [21], which initializes the offset value to zero, and the weights for each layer are determined from the normal distribution formula.

For computing the loss and the necessary derivatives in TF, automatic differentiation was used. During the forward pass of the neural network, TF creates a computational graph that stores the operations performed on the tensors and their sequence. In the automatic differentiation step, the chain rule is applied to compute the derivative of the loss function with respect to the weight coefficients and biases based on the computational graph obtained during the forward pass.

The neural networks were trained on an Nvidia GeForce RTX 3080 Ti Laptop graphics card.

The finite element method is implemented using the FEniCS library using piecewise linear basis functions for the approximation.

The Crank-Nicolson method, defined by (3), was used for time discretisation. This method is certainly numerically stable for diffusion equations and beyond. It has a second-order accuracy with respect to Δt .

$$u_{n+1} = u_n + \frac{\Delta t}{2} (F_n + F_{n+1}), \quad (3)$$

where Δt is the value of the time discretisation step, u^n is the approximate solution at time $t_n = \frac{n}{N}$, $n \in \{0, 1, 2, \dots, N\}$ is the time step number, N is the number of time steps, u^0 is the initial condition, F_n is the right-hand side of the non-stationary equation at time t_n .

3. Strengths and Limitations

Advantages of PINNs:

- Flexible problem setup. Changing the governing equations or boundary conditions is straightforward, which makes adapting to new problems easy;
- Geometry-agnostic modeling. PINNs can handle arbitrarily complex domains without requiring a mesh, simplifying the treatment of intricate geometries. Training points density must still reflect local solution scales; poor sampling can harm accuracy;
- Built-in physics knowledge. By embedding the differential equations directly into the network's loss, PINNs leverage a priori knowledge of the problem structure, improving both accuracy and efficiency;
- Multidimensional and time-dependent capability. A single PINN formulation can solve stationary and non-stationary, low- or high-dimensional problems without resorting to separate time-discretization schemes;
- Advanced optimization. They can exploit state-of-the-art training algorithms and regularization techniques to accelerate convergence;
- Interoperability. PINNs integrate seamlessly with other machine-learning algorithms and data-driven methods.

Disadvantages of PINNs:

- Risk of local minima. Like all gradient-descent-based approaches, PINNs can become trapped in local minima, yielding suboptimal solutions;
- Hyperparameter sensitivity. Selecting the optimal network architecture, learning rate, weighting of loss terms, etc., often demands extensive tuning and computational effort;
- No guaranteed convergence. There is no general convergence guarantee for PINN training—particularly on highly nonlinear problems and with multiple local minima, so gradient-based optimization may never find the true (global) solution.
- High computational cost. Training PINNs on large-scale or high-dimensional tasks can require substantial CPU/GPU resources, which may limit their practical use in some settings.

Recent theory proves convergence of PINNs only in narrow settings—e.g., linear second-order elliptic and parabolic PDEs with smooth coefficients (Hölder-continuous) and infinite points samples [22]. For the general nonlinear, stiff, or multimodal (loss landscapes containing many distinct local minima or several alternative solutions) case, no global convergence proof exists; instead, empirical studies reveal gradient-flow pathologies (vanishing, exploding, or mutually orthogonal gradients) that stall training in local minima [23], failure modes in which the network fails to capture even basic physical scenarios [24], and a sharp drop in success probability as PDE order or dimensionality rises [25, 26]. Attempts to fix these issues—loss-term re-weighting [27], curriculum sampling [28] or symmetry [29] — improve robustness but still lack rigorous guarantees. Comprehensive convergence theory therefore remains an open research problem, and practitioners must validate results against reference solvers or a posterior error estimators [30, 31].

4. Results and discussions

4.1. Problem 1

Let us consider the boundary value problem for stationary diffusion:

$$\Delta u = 0, x \in \Omega = (0; 1) \times (0; 1), \quad (4)$$

$$u = 0 \text{ on } \Gamma_1, \quad (5)$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_2, \quad (6)$$

$$u = 1 \text{ on } \Gamma_3, \quad (7)$$

$$\frac{\partial u}{\partial n} = u \text{ on } \Gamma_4. \quad (8)$$

Table 1

Selected hyperparameters of the neural network for the problem (4)-(8)

Learning speed	Nodes on one axis	Hidden layers	Number of neurons in one layer	Stopping criterion (loss)	Activation function
1e-2	11	3	10	1e-3	tanh

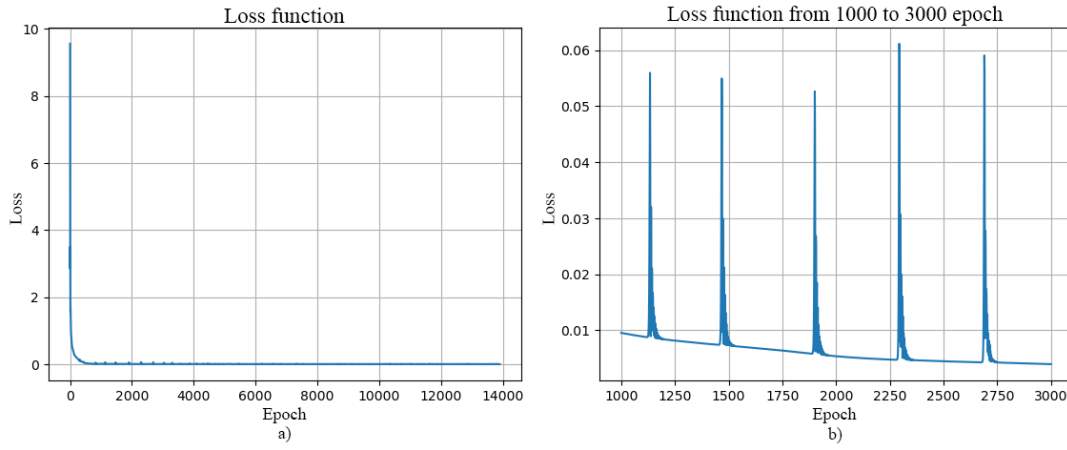


Figure 2: a) The loss function up to the final epoch. b) The loss function from the 1000th to 3000th epoch.

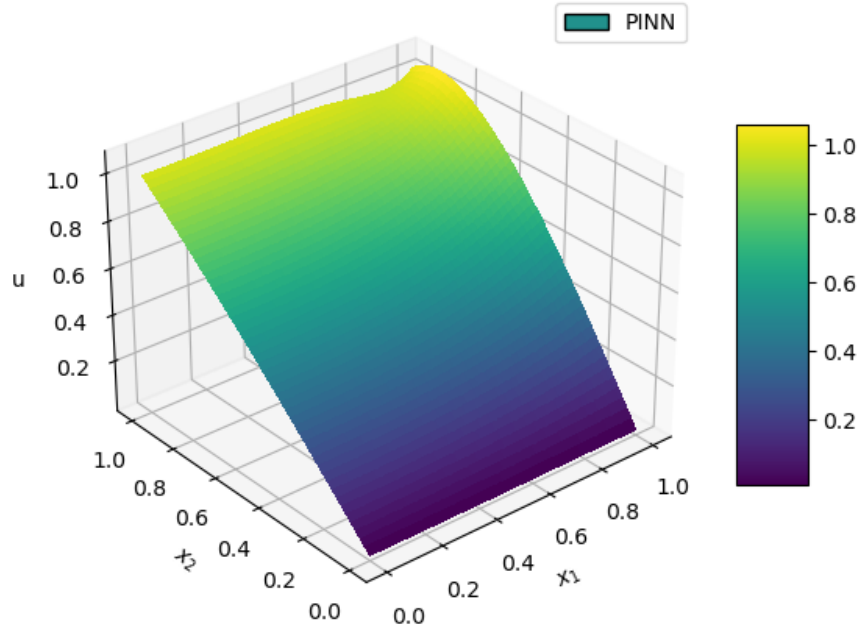


Figure 3: The approximate solution of (4)-(8) obtained by PINN after 13900 epochs.

Figure 2 illustrates that after 1000 epochs the NN achieved loss as low as 0.01, compared to initial value around 9. Also, in Figure 2 b) can be seen how the network tries to get out of local minimum to get a better solution, closer to the global minimum. Figure 3 illustrates solutions obtained by PINN after 13900 epochs of training. This neural network has 261 parameters to train. Training of the PINN took 3 min 49 sec to achieve the loss given in Table 1. The average training time is 4 minutes and 33 seconds among the 5 training sessions. The deviation of the average time from the minimum time in the sample data is 53s, and from the maximum training time is 89s.

To balance the influence of different components of the loss function, the following coefficients were chosen in (2): for Robin's condition on the right boundary Γ_4 , $\lambda=2$, all other λ coefficients are equal to one.

For the FEM, linear rectangular finite elements on a 10x10 grid were used.

Figure 4 and Figure 5 show that the approximate solution corresponds to the error in l_∞ norm of 2.92 % calculated in Table 2. In Figure 5, we can clearly see that NN, which is a non-linear function, tries to approximate a line. Therefore, we will always observe some deviations, even as the number of epochs or the number of training points increases.

The relatively low relative error in the l_∞ norm (2.92 %) indicates that the PINN approximates the stationary diffusion problem with high accuracy, comparing favorably with the FEM results. The careful selection of loss coefficients, particularly using $\lambda=2$ for the Robin boundary condition, appears to be instrumental in balancing the contribution of various loss components. This balance is essential to achieve a stable convergence and accurate solution. Overall, these findings suggest that

PINN is a viable alternative for solving such boundary value problems, especially when rapid adjustments to the problem setup are required.

Table 2

Error between the approximate solutions obtained by PINN and FEM for problem (4)-(8)

Type of error	Error value (13900 epoch)	Error value (6000 epoch)	Error value (3000 epoch)
Mean square error	6.86e-06	5.12e-05	2.62e-05
Relative error in the l_∞ norm	2.92 %	4.2 %	5.24 %
Relative error in the l_2 norm	0.41 %	1.12 %	0.81 %

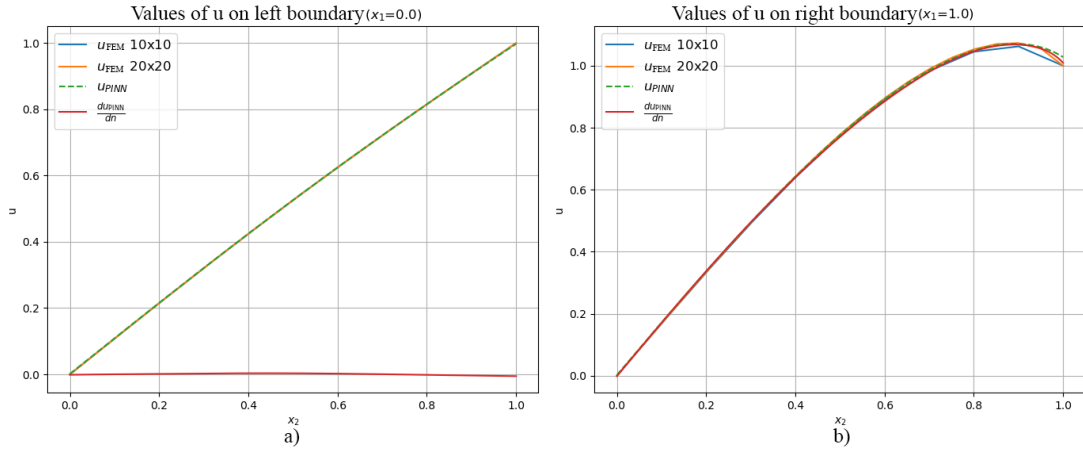


Figure 4: a) The approximate solution of (4)-(8) obtained by PINN and FEM for Γ_2 . b) The approximate solution of (4)-(8) obtained by PINN and FEM for Γ_4 .

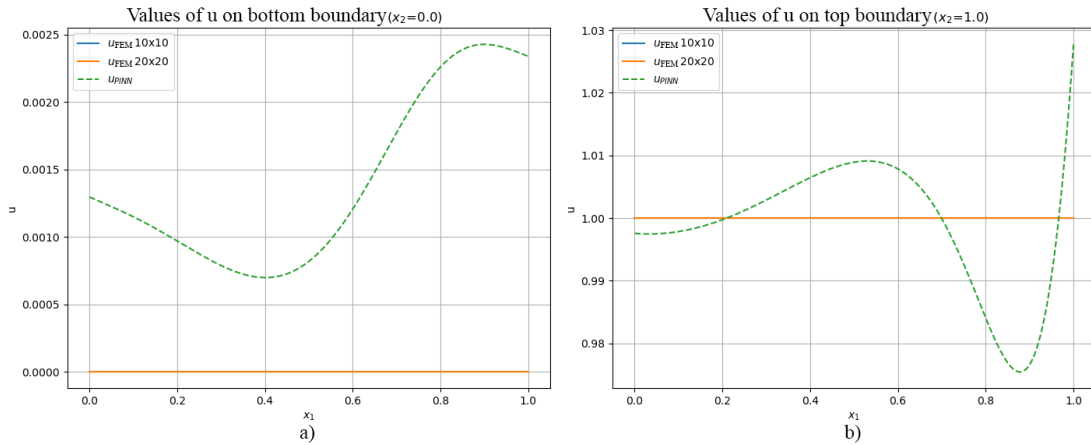


Figure 5: a) The approximate solution of (4)-(8) obtained using PINN and FEM for Γ_1 . b) The approximate solution of (4)-(8) obtained using PINN and FEM for Γ_3 .

4.2. Problem 2

Let us consider a one-dimensional non-stationary problem for the heat conduction equation:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, (t, x) \in (0; 0.2) \times (0; 1), \quad (9)$$

$$u(0, t) = 0, \quad (10)$$

$$u(1, t) = 0, \quad (11)$$

$$u(x, 0) = \sin(\pi x) - \sin(2\pi x) + \sin(3\pi x). \quad (12)$$

The analytical solution for (9)-(12) is obtained by applying the method of separation of variables:

$$u(x, t) = e^{-\pi^2 t} \sin(\pi x) - e^{-(2\pi)^2 t} \sin(2\pi x) + e^{-(3\pi)^2 t} \sin(3\pi x). \quad (13)$$

This neural network has 1981 parameters to train. Training of the PINN took 5 min 42 sec to achieve the loss given in Table 3. The average training time is 4 minutes and 17 seconds among the 5 training sessions. The deviation of the average time from the minimum time in the sample data is 77s, and from the maximum training time is 95s.

To balance the influence of different components of the loss function, the following coefficients were chosen in (2): for the initial condition (when $t=0$) $\lambda=5$, for the Dirichlet conditions on the left and right boundaries $\lambda=50$, and for the equation $\lambda=0.7$.

For the FEM, linear rectangular finite elements on a 25x25 grid were used.

Figure 8 and Figure 9 show that the approximate solution corresponds to the errors calculated in Table 4 (the error between the PINN and the analytical solution in the l_∞ norm is 1.17 %). Figure 9 once again shows, that we approximate a line with the non-linear function in form of the neural network and will always get some deviations for approximating linear function. Also, in Table 4, we can see the error values for different numbers of epochs.

Figure 6 shows a graph of the loss function up to the final epoch, as well as from the 1000th to 3000th epoch. From Figure 6 a) after 4000 epoch small spikes of values of the loss functions are observed, as the optimizer tries to get out of local minimum to get closer to the global minimum. Figure 7 shows the approximate solution obtained by the PINN after 12441 training epochs.

Table 3

Selected hyperparameters of the neural network for the problem (9)-(12)

Learning speed	Nodes on one axis	Hidden layers	Number of neurons in one layer	Stopping criterion (loss)	Activation function
5e-4	25	3	30	2e-2	tanh

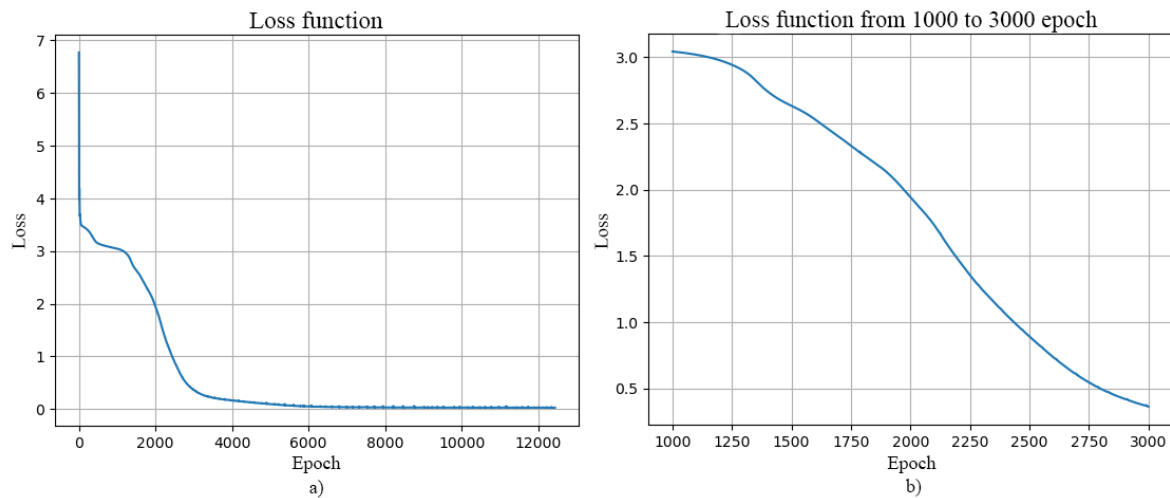


Figure 6: a) The loss over training epochs for the problem (9)-(12) up to the last 12441st epoch. b) The loss over training epochs for the problem (9)-(12) from the 1000th to 3000th epoch.

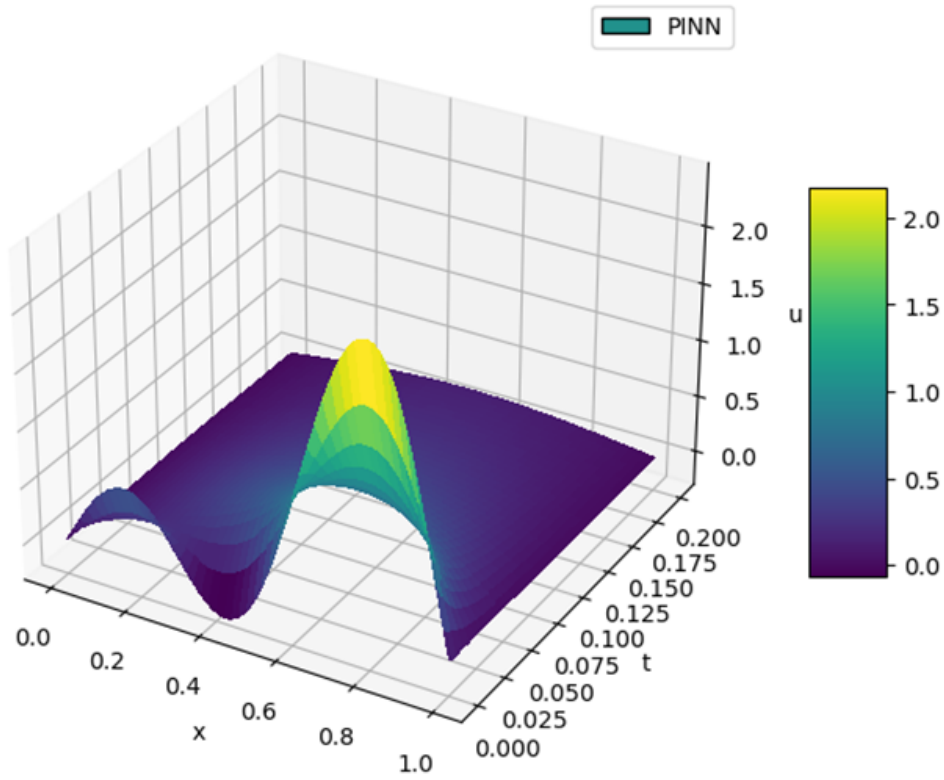


Figure 7: The approximate solution of (9)-(12) obtained by PINN after 12441 epochs.

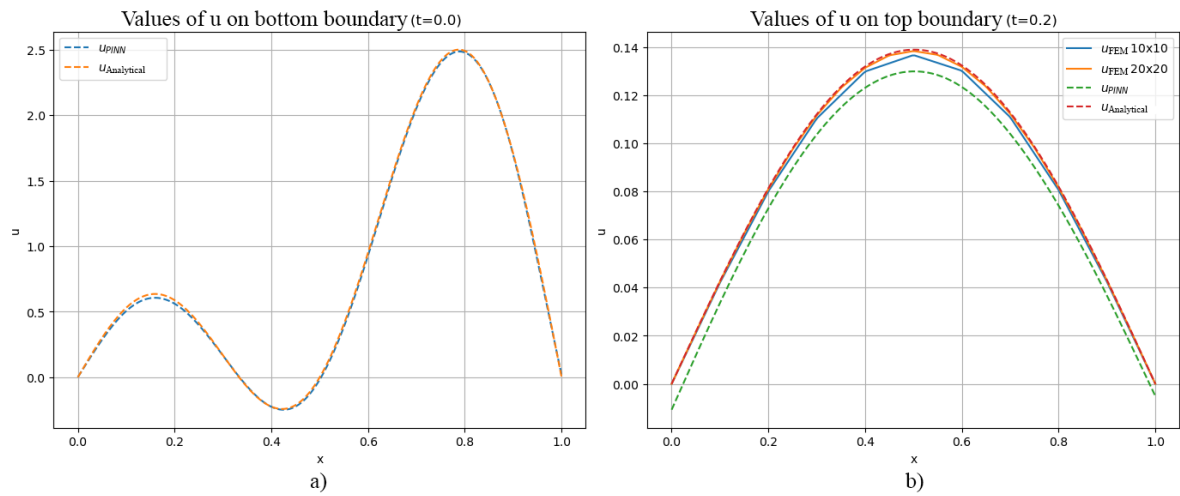


Figure 8: a) The approximate solution of (9)-(12) using PINN and the initial condition (12) at $t=0$. b) The approximate solution of (9)-(12) using PINN, FEM and the analytical solution at the final time $t=0.2$.

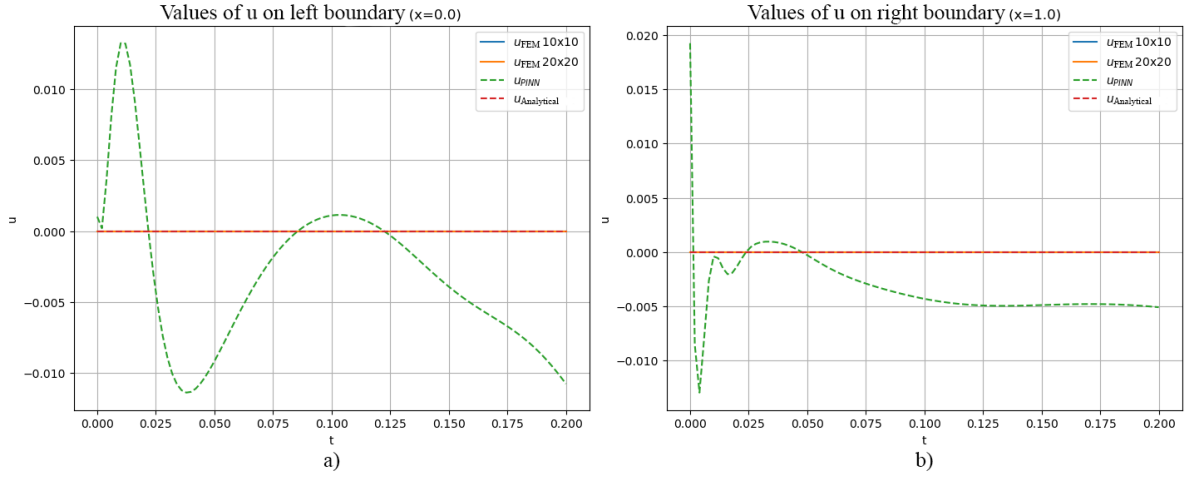


Figure 9: Figures a) and b) show the graphs of the approximate solution of (9)-(12) using PINN and FEM on the left and right boundaries.

Table 4

Error between the approximate solutions obtained by PINN and the analytical solution (13) for the problem (9)-(12)

Type of error	Error value (12441 epoch)	Error value (6000 epoch)	Error value (3000 epoch)
Mean square error	4.91e-05	5.4	1.81
Relative error in the l_∞ norm	1.17 %	98.7 %	1e02 %
Relative error in the l_2 norm	1.69 %	88.8 %	83.1 %

Table 5

Error between the approximate solutions obtained by PINN, FEM and the analytical solution (13) for the problem (9)-(12). Separate training session from results from Table 4

Type of error	PINN and FEM	PINN and analytical	FEM and analytical
Mean square error	6.43e-05	7.01e-05	1e-05
Relative error in the l_∞ norm	1.72 %	1.72 %	1.07 %
Relative error in the l_2 norm	1.95 %	2.03 %	0.76 %

The error trends presented in Table 4 show that longer training (more epochs) yields more accurate approximations, as seen by the decrease in mean square error with increased training. The relatively low error in the l_∞ norm (1.17 % at 12441 epochs) underscores the robustness of the PINN approach for this non-stationary heat conduction problem. Furthermore, the carefully chosen loss coefficients — particularly the higher weights for the boundary conditions — appear to contribute to the stable convergence of the solution. Moreover, Table 5 — presenting results from an independent training session that compares PINN with FEM and the analytical solution — shows that the relative errors are consistently low (approximately 1.72 % for PINN against both FEM and the analytical model and 1.07 % for FEM against the analytical solution). These results validate the effectiveness of PINN in capturing the dynamics of time-dependent problems while also highlighting the trade-off between training time and solution accuracy.

4.3. Problem 3

Definition. Let there be two closed connected domains $X, U \in \mathbb{R}^2$ such that $X \subsetneq U$, $\partial X \cap \partial U = \emptyset$. Then, a doubly connected domain is a domain D such that: $D = \overline{U} \setminus \overline{X}$.

Let $D \subset \mathbb{R}^2$ be a limited doubly connected domain with sufficiently smooth boundaries $\Gamma_1, \Gamma_2 \in C^2$, whose parametric definitions are given by (14) and (15).

$$\Gamma_1 = \{x_1(\varphi) = (2 \cos(\varphi), 2 \sin(\varphi)), \varphi \in [0, 2\pi]\}, \quad (14)$$

$$\Gamma_2 = \{x_2(\varphi) = (5 \cos(\varphi), 5 \sin(\varphi)), \varphi \in [0, 2\pi]\}. \quad (15)$$

Let us consider the problem of stationary heat conduction in a doubly connected domain D:

$$\Delta u = 0 \text{ in } D \quad (16)$$

and boundary conditions

$$u = x \text{ on } \Gamma_1, \quad (17)$$

$$u = 0 \text{ on } \Gamma_2. \quad (18)$$

This neural network has 1981 parameters to train. A pseudo-random number generator was used to generate the input data. The neural network was trained for 4 minutes and 20 seconds to achieve the losses given in Table 6.

To balance the influence of different components of the loss function, the following coefficients were chosen in (2): for the Dirichlet conditions on the left and right boundaries $\lambda = 50$ and for the equation $\lambda = 0.7$.

For the FEM, 197 linear triangular finite elements were used.

Figure 10 shows how the loss function decreases rapidly at the initial learning epochs. Figure 11 shows the approximate solution obtained by the PINN after 10611 training epochs. Graphs on the boundaries and errors for the entire domain for the last epoch and intermediate ones are shown in Figure 11 and Table 7, respectively. Figure 12 illustrates the values of the analytical solution and the approximate solution obtained by PINN after training and on intermediate epochs. As can be seen from the approximate solution in Figure 12 b) on intermediate epochs, u is approximated by a nonlinear function (NN consists of tanh activation functions and linear combinations), and the more we train NN, the closer it will get to a straight line, but will always contain some deviations.

Table 6

Selected hyperparameters of the neural network for the problem (16)-(18)

Learning speed	Number of internal points and on the border	Hidden layers	Number of neurons in one layer	Stopping criterion (loss)	Activation function
5e-4	500 and 50	3	30	1e-3	tanh

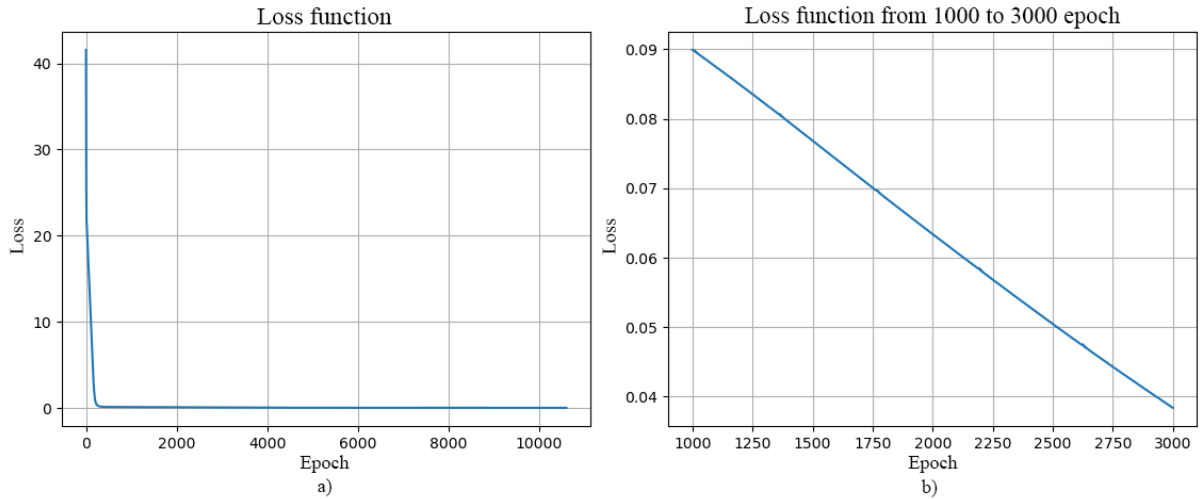


Figure 10: a) The loss over training epochs for the problem (16)-(18) up to the last 10611 epoch. b) The loss over training epochs for the problem (16)-(18) from the 1000th to 3000th epoch.

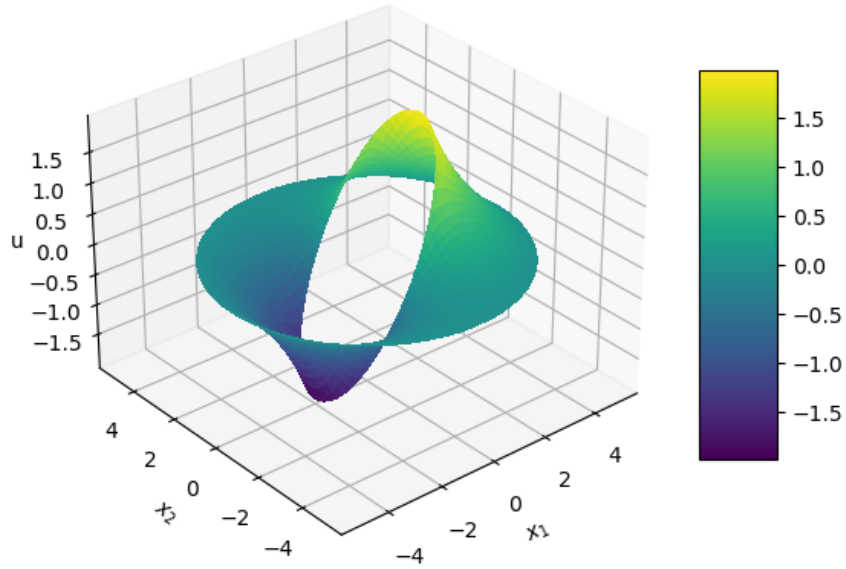


Figure 11: The approximate solution of (16)-(18) obtained by PINN after 10611 epochs.

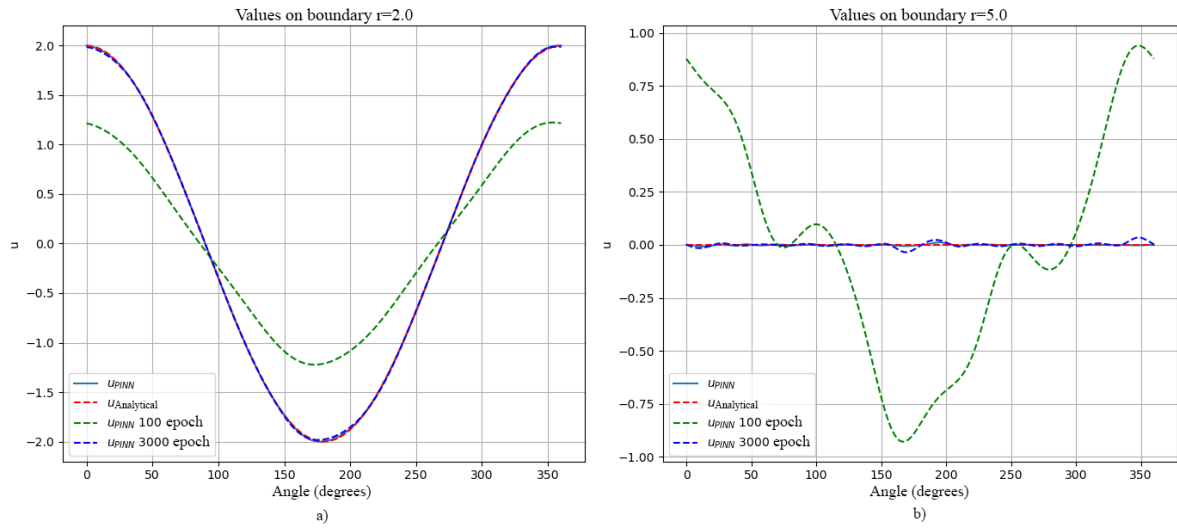


Figure 12: a) The approximate solution of (16)-(18) obtained by PINN (10611 epochs, 3000 epochs, 100 epochs) and the analytical value for Γ_1 . b) The approximate solution of (16)-(18) obtained by PINN (10611 epochs, 3000 epochs, 100 epochs) and the analytical value for Γ_2 .

Table 7

Error between the approximate solutions obtained by PINN and FEM solution for the problem (16)-(18)

Type of error	Error value (10611 epoch)	Error value (6000 epoch)	Error value (3000 epoch)
Mean square error	0.613	0.666	0.75
Relative error in the l_∞ norm	2.26 %	9.54 %	77.4 %
Relative error in the l_2 norm	1.95 %	10.2 %	63.8 %

The rapid decrease in loss during the early training epochs (as seen in Figure 10) demonstrates that the PINN quickly adapts to the challenges posed by the doubly connected domain. Despite the added complexity due to the domain geometry, the PINN achieves a competitive performance relative to the FEM approach, as indicated by the error metrics in Table 7. The gradual improvement in accuracy with extended training epochs underscores the importance of adequate training, especially for complex geometries. These results support the potential of PINNs for application in more intricate

heat conduction problems and suggest that further research into optimization strategies for training time reduction could yield even better performance.

5. Conclusions

This study set out to determine whether a compact feedforward PINN can reproduce finite-element (FEM) accuracy for linear, second-order parabolic and elliptic PDEs while accommodating mixed boundary conditions or non-rectangular geometries. For each problem: 2-D Laplace on a square, the same equation on a doubly-connected domain, and a 1-D non-stationary heat-conduction problem—the network depth and width were tuned empirically to minimize loss within a fixed training time. The resulting models achieved l_∞ errors of 2.92 %, 2.26 % and 1.17 %, respectively, after 4–6 minutes of training on an RTX 3080 Ti Laptop GPU, matching FEM accuracy while eliminating mesh generation and allowing rapid modification of governing equations or boundary conditions. Because time is treated as an additional input, the non-stationary case required no explicit time discretization scheme, further simplifying implementation. These results indicate that, for canonical diffusion problems in one and two dimensions with constant coefficients, a mesh-free PINN can match FEM accuracy when development flexibility outweighs computation time. These empirical results apply to the constant-coefficient cases we tested; however, convergence proofs [22] hold more generally for linear second-order parabolic and elliptic PDEs with Hölder-continuous (smooth) coefficients, so the theoretical foundation extends beyond the constant-coefficient setting.

The experiments also outline clear performance boundaries. Training remains markedly slower than FEM for comparable resolution, and optimization is sensitive to the relative weighting of interior, boundary and initial residuals. A broader survey of the literature shows that standard PINNs can struggle on more challenging PDEs: convergence proofs are still absent for nonlinear, stiff, high-frequency, or higher-order cases, and empirical work reports gradient-flow pathologies: vanishing, exploding, or mutually orthogonal gradients—that stall training and become more severe in higher spatial dimensions. Although our experiments focused on constant-coefficient diffusion problems, existing convergence theorems guarantee PINN convergence for any linear second-order parabolic or elliptic PDE with Hölder-continuous coefficients [22], and these theoretical and empirical findings together indicate that today’s standard PINNs remain most reliable on modest-scale, well-conditioned diffusion examples.

Future research should slash training cost and enable high-dimensional scalability; extend PINNs to nonlinear, stiff, coupled, and stochastic PDEs; and rigorously validate their advantages on engineering and physical-science problems where meshing or classical solvers are prohibitive. Achieving these goals will elevate PINNs from promising prototypes to versatile, production-ready solvers.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

- [1] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv preprint arXiv:1711.10561, 2017. doi:10.48550/arXiv.1711.10561.
- [2] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, arXiv preprint arXiv:1711.10566, 2017. doi:10.48550/arXiv.1711.10566.
- [3] S.R. Vadyala, S.N. Betgeri, Physics-informed neural network method for solving one-dimensional advection equation using PyTorch, arXiv preprint arXiv:2103.09662, 2021. doi:10.48550/arXiv.2103.09662.
- [4] C.C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018, 497 p. doi:10.1007/978-3-319-94463-0.

- [5] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. doi:10.1016/0893-6080(89)90020-8.
- [6] D. Alianakh, Implementation of a physics-informed neural network for solving partial differential equations, in: Proc. of the International Student Scientific Conference on Applied Mathematics and Computer Science (ISSCAMCS-2023), Lviv, May 4–5, 2023, pp. 64–67. [In Ukrainian]. URL: <https://ami.lnu.edu.ua/wp-content/uploads/2023/06/ISSCAMCS-2023.pdf>.
- [7] D. Alianakh, I. Dyyak, M. Selivanov, Development of deep learning for mathematical physics problems, in: Proc. of Modern Problems of Applied Mathematics and Computer Science (APAMCS-2023), Lviv, November 7–9, 2023, pp. 76–78. [In Ukrainian]. URL: https://lnueduua-my.sharepoint.com/:b:/g/personal/petro_venhershky_lnu_edu_ua/EWUrULRyg9cxHvbPBBphmdbcBXPg3nHiGuhcjuIj7NQ118g?e=Io2B8o.
- [8] Y. Zong, Q. He, A.M. Tartakovsky, Physics-Informed Neural Network Method for Parabolic Differential Equations with Sharply Perturbed Initial Conditions, arXiv preprint arXiv:2208.08635, 2022. doi:10.48550/arXiv.2208.08635.
- [9] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.
- [10] E. Kharazmi, Z. Zhang, G. E. Karniadakis, VPINNs: Variational Physics-Informed Neural Networks for Solving Partial Differential Equations, arXiv preprint arXiv:1912.00873, 2019. doi:10.48550/arXiv.1912.00873.
- [11] Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, Y. Liu, Kolmogorov–Arnold-Informed Neural Network: A Physics-Informed Deep Learning Framework for Solving Forward and Inverse Problems Based on Kolmogorov–Arnold Networks, arXiv preprint arXiv:2406.11045, 2024. doi:10.48550/arXiv.2406.11045.
- [12] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, *Computer Methods in Applied Mechanics and Engineering*, vol. 389, art. 114333, 2021. doi:10.1016/j.cma.2021.114333.
- [13] N. Zobeiry, K. D. Humfeld, A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications, *Engineering Applications of Artificial Intelligence*, vol. 101, art. 104232, 2021. doi:10.1016/j.engappai.2021.104232.
- [14] F. Sahli Costabal, S. Pezzuto, P. Perdikaris, Δ -PINNs: Physics-informed neural networks on complex geometries, arXiv preprint arXiv:2209.03984, 2022. doi:10.48550/arXiv.2209.03984.
- [15] R. J. Gladstone, M. A. Nabian, N. Sukumar, A. Srivastava, H. Meidani, FO-PINNs: A First-Order formulation for Physics-Informed Neural Networks, arXiv preprint arXiv:2210.14320, 2022. doi:10.48550/arXiv.2210.14320.
- [16] Q. Yang, Y. Yang, T. Cui, Q. He, FDM-PINN: Physics-informed neural network based on fictitious domain method, *Int. J. Comput. Math.* 100 (2022) 1. doi:10.1080/00207160.2022.2128674.
- [17] O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, Butterworth-Heinemann, 2013, 756 p.
- [18] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover Publications, 2000, 704 p.
- [19] D. Harrison, A Brief Introduction to Automatic Differentiation for Machine Learning, arXiv preprint arXiv:2110.06209v2, 2021. URL: <https://arxiv.org/abs/2110.06209v2>.
- [20] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proc. of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, May 7–9, 2015. doi:10.48550/arXiv.1412.6980.
- [21] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256. URL: <https://proceedings.mlr.press/v9/glorot10a.html>.
- [22] Y. Shin, J. Darbon, G. E. Karniadakis, On the convergence of physics-informed neural networks for linear second-order elliptic and parabolic type PDEs, arXiv preprint arXiv:2004.01806, 2020. doi:10.48550/arXiv.2004.01806.
- [23] P. Rathore, W. Lei, Z. Frangella, L. Lu, M. Udell, Challenges in Training PINNs: A Loss Landscape Perspective, arXiv preprint arXiv:2402.01868, 2024. doi:10.48550/arXiv.2402.01868.

- [24] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, M. W. Mahoney, Characterizing possible failure modes in physics-informed neural networks, in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 21416–21427. doi:10.48550/arXiv.2109.01050.
- [25] Z. Hu, K. Shukla, G. E. Karniadakis, K. Kawaguchi, Tackling the curse of dimensionality with physics-informed neural networks, *Neural Networks*, vol. 176, p. 106369, May 2024. doi:10.1016/j.neunet.2024.106369.
- [26] C. H. Song, Y. Park, M. Kang, How does PDE order affect the convergence of PINNs? in *The 38th Annual Conference on Neural Information Processing Systems*, 2024. URL: <https://openreview.net/forum?id=8K6ul0hgtC>.
- [27] Q. Liu, M. Chu, N. Thuerey, ConFIG: Towards Conflict-free Training of Physics-Informed Neural Networks, arXiv preprint arXiv:2408.11104, 2024. doi:10.48550/arXiv.2408.11104.
- [28] M. Münzer, C. Bard, A Curriculum-Training-Based Strategy for Distributing Collocation Points during Physics-Informed Neural Network Training, *NeurIPS Workshop on Machine Learning for the Physical Sciences (ML4PS)*, 2022; arXiv preprint arXiv:2211.11396, Nov. 2022. doi:10.48550/arXiv.2211.11396.
- [29] Z.-Y. Zhang, H. Zhang, L.-S. Zhang, L.-L. Guo, Enforcing continuous symmetries in physics-informed neural network for solving forward and inverse problems of partial differential equations, *Journal of Computational Physics*, vol. 492, p. 112415, Nov. 2023. doi:10.1016/j.jcp.2023.112415.
- [30] N. Doumèche, G. Biau, C. Boyer, Convergence and error analysis of PINNs, arXiv preprint arXiv:2305.01240, 2023. doi:10.48550/arXiv.2305.01240.
- [31] M. Zeinhofer, R. Masri, K.-A. Mardal, A Unified Framework for the Error Analysis of Physics-Informed Neural Networks, arXiv preprint arXiv:2311.00529, Nov. 2023. doi:10.48550/arXiv.2311.00529.